ISSN 1330-3651
UDC/UDK 004.424:338.242

MANAGING REQUIREMENTS IN MARKET-DRIVEN SOFTWARE PROJECT: AGILE METHODS VIEW

Deepti Mishra, Alok Mishra

Subject review

Time-to-market and insufficient initial requirements are two significant challenges that make managing requirements for market-driven software projects different from custom-made software projects. These challenges can be resolved by using agile software development methodologies for market-driven software development as agile methods put emphasis on a dynamic approach for requirement engineering which works closely with an iterative release cycle. In this study, dynamic requirement engineering approach of Agile methods was used for the successful implementation of market-driven complex software project.

Keywords: *requirements engineering, market-driven software, Agile Methods, Agile Software Development (ASD)*

Organiziranje potreba u tržišno orijentiranom softverskom projektu u skladu s Agilnim metodama

Pregledni članak

Vrijeme do plasiranja na tržište i nedovoljno početno saznanje o potrebama dva su značajna problema po kojima se organiziranje potreba kod projekata razvijanja softvera za tržište razlikuje od projekata za softver po narudžbi. Ti se problemi mogu riješiti primjenom agilnih metodologija razvoja softvera kad se radi o tržišno orijentiranom softveru budući da Agilne metode naglašavaju dinamički pristup organiziranju potreba usko povezan s iterativnim ciklusom isporuke. U ovom radu, dinamički pristup Agilnih metoda definiranju potreba koristi se za uspješnu implementaciju složenog tržišno orijentiranog softverskog projekta.

Ključne riječi**:** *definiranje potreba, tržišno orijentirani softver, agilne metode, agilni razvoj softvera*

1 Introduction
Uvod

Increased competitive pressures are driving compressed schedules to enable the organization to meet time-to-market goals, while, at the same time, teams face daunting budgetary challenges. Surviving, in this context, is no longer sufficient, and organizations must fundamentally change the way they developed systems to thrive and grow [1]. Software development is a series of resource-limited, goal-directed cooperative games of invention and communication [2]. Requirements Engineering (RE) activities are considered critical to any software development process. It has been recognized that problems associated with requirements area are among the major reasons for software project failures [3, 4]. The effort to explore and refine RE has increased in the last years, as observed by Nuseibeh [5] and Cheng and Atlee [6] in their studies about the present and the future of RE. However, there are still a few studies about how real life agile projects identify and manage the customer requirements [7].

Requirement engineering for market-driven software development is different from customer specific software development. In Market-driven projects, requirement gathering is difficult as there are no distinct and defined sets of users [8]. There are potential users, an imagined group of people who may fit into the profile of an intended product user [9], who can help in gathering some requirements. Often, requirements are invented by developers [10], based on strategic business objectives, domain knowledge and a product vision. All requirements cannot be known in advance before construction begins. In fact, there is a constant flow of requirements from various sources and these requirements need to be prioritized and managed during software development life cycle. This is similar to the requirement engineering phase of Agile methods. Agile methods have been demonstrated to enhance customer value, improve organizational morale, and provide step improvement in product quality [11]. In Agile approach, development of requirements specifications is conceived as an incremental process, in which the stakeholders successively add requirements until attaining the specifications of the desired system [12]. Orr [13] suggests that it is possible to combine requirements and agile development by using up-to-date hardware and sophisticated graphical software. Ambler [14] describes an agile approach to modeling requirements, utilizing approaches such as planning game of Extreme Programming (XP) and the Scrum Methodology. Leffingwell and Widrig [15] argue for an agile requirements technique that is based on use-case specifications. Nawrocki et al. [16] propose a way in which documented requirements can be introduced into XP through the use of automated tools, the Web and on-line documentation.

When developing software for a market place rather than bespoke software for a specific customer, short time-to-market is very important [17, 18]. It is important so as not to lose the market share to competitors and in case of probable delays, only high priority requirements are implemented in the current release. Low priority requirements are excluded from the current release and implemented in subsequent releases. Therefore, market-driven software products are often developed in several consecutive releases. This is in sync with the philosophy of Agile Methods which states software should be developed in an incremental and iterative way with high priority requirements to be included in initial releases and working software is seen as a sign of progress.

Active participation of the user during development is one of the highly important principles of Agile methods. Similarly, in order to succeed and capture the market with their market-driven software, organizations must have some means to get customer feedback early in the

development process and thus minimize the risk of wasting valuable development effort because of ambiguous and incomplete specifications.

Since Agile is a relatively young process model, there are few studies with relevant results about the elicitation and management of requirements [7]. Requirement elicitation activity intends to identify and understand customer needs. In agile approaches development tasks are not centered in a complete and well-defined set of requirements. User needs are incrementally elicited. Some of the open issues in agile methodologies concern elicitation of non-functional requirements and requirements documentation tasks [7]. Furthermore, a recent survey by Vijayasarathy and Turk [3] points out that currently inadequate project requirements and instability of requirements are among the important limitations of agile methods. Agerfalk and Fitzgerald [19] have observed that “practice is ahead of research” in this area. Rajlich [20] further supported this that agile software development brings a host of new topics into software engineering research, that there exists a "backlog of research tasks". Dyba and Dingsoyr [21] found in their review that there is need to increase both the number and quality of studies on agile software development. They further argued that agile project management methods, such as Scrum, which are popular in industry, warrant further attention. Rodriguez et al. [7] recently observed that there are still few studies about how real life agile projects identify and manage customer requirements.

We developed market-driven software (supply chain management software) with the motivation of Agile methods and the requirement engineering phase was done in an iterative way. We gathered requirements by conducting several sessions of interviews and workshops. Also, feedback from initial release helped in refining old requirements and gathering new ones. In this paper, we discuss how the application of agile methods for market-driven software development resulted in the successful implementation of supply chain management software.

The remainder of the paper is organized as follows: In section 2, we present a review on requirements engineering issues for market-driven software development and agile software development. In section 3, an industrial case study is illustrated. Section 4 summarizes discussion and lessons learned. Finally, the paper concludes with conclusion and future research directions in this area.

2 Literature Survey
Pregled literature

Karlsson et al. [9] investigated current practices and challenges for market-driven requirement engineering in Swedish software development organizations in order to increase the understanding of the area of market-driven requirement engineering. They found many problems, some of them unique for market-driven projects and not applicable to customer-specific software projects. Therefore, requirement engineering methods to develop customer-specific software may not be enough to support requirement engineering for market-driven software projects. This is also supported by Potts [10] that requirement engineering models and methods to develop bespoke software are not suitable for market-driven software development. He suggested some alternatives that would address these shortcomings for research and consulting communities.

In market-driven projects, eliciting requirements is difficult as there are no defined users but some potential customers. These requirements should be prioritized also as time-to-market is a key to success and also for capturing the market and all gathered requirements may not be implemented within a specified time limit. Only high priority requirements are implemented in the current release and low priority requirements are left to be included in subsequent releases. Therefore, successful requirement engineering process for market-driven software development projects must have the ability to extract requirements from different sources. These different sources may give conflicting requirements so there must be some way to resolve conflicts among requirements and then prioritize and manage them. Lueke [22] presented a Structured Brainstorming and Evaluative Survey Technique (SBEST) for discovering, systematically gathering, prioritizing and implementing marketplace wants and needs while paying attention to any competition. SBEST yields the attributes of the ideal solution from the market's point of view. However, the process does not yield specifications for development of a product or service. Yeh [23] presented a market-driven requirement management process (REQUEST) that transforms systematically the many "voices of Customers" through various stages to a set of plan candidates by means of analysis, validation, and prioritization. It tracks and relates original requirements to plan items and vice versa. Tuunanen and Rossi [24] developed a new RE method based on Critical Success Chain (CSC) method that includes top-down approach of planning and participation of information systems customers to get rich information. They extended CSC with customer segmentation and lead user concepts from marketing. They also constructed a support environment within Metaedit+ Meta CASE tool to present and manage requirements. Regnell [25] presented an industrial case study where a distributed prioritization process is proposed, observed and evaluated. A major objective of the distributed prioritization is to gather and highlight the differences and similarities in the requirement priorities of the different market segments. Various charts are proposed to present visually the disagreement between stakeholders and differences in satisfaction with a certain priority decision. These charts are intended to be used as decision support when determining what to implement in the coming release of a software package.

Sawyer et al. [17] pointed out that time-to-market is the overriding constraint for market-driven software development projects. When the project falls behind schedule, the preferred solution is to concentrate on meeting the most critical requirements and releasing the product on time. Other features can be added in later releases. Also, new requirements will also emerge when real users start using the software. So, requirement engineering process for market-driven software development has to be dynamic and must work closely with an iterative release cycle. Sawyer et al. [17] synthesized a number of good practices for requirement engineering for packaged software.

Also, there is a need to handle congestion in the requirement engineering process for market-driven software development which may occur when short time-to-market is combined with the rapid arrival of new requirements from many different sources. Eliminating duplicity of requirements helps in dealing with congestion. Natt och Dag et al. [26] presented empirical evaluations of the benefits of automated similarity analysis of textual

requirements, where existing information retrieval techniques are used to statistically measure requirements similarity. Host et al. [27] modeled a specific requirement management process (REPEAT) using discrete event simulation and the parameters of the model were estimated based on interviews with people from the specific organization where the process is used. Their aim was to investigate if simulation can help in exploring bottlenecks and overload situations in requirement engineering processes and to find changes to the process that may remove bottlenecks.

To compete in the market, the product should contain features or functions that do not exist in other similar software so developers tend to put more effort into inventing and implementing new functional features that are expected to improve the product but these new functions are useful only if users can use them easily. So, adding new functions is not only important but these newly added functions must be also usable. This is supported by Natt och Dag et al. [28] that although developers rely heavily on the number and the existence of new features, usability is recognized as a competitive advantage on its own. They presented results and experiences of an industrial case study that employs two known usability evaluation methods (a questionnaire and a heuristic evaluation) at a market-driven software development company inexperienced in usability.

The requirement engineering phase of our project did not rely on any particular method mentioned above. We tried to learn lessons from all the above mentioned studies and used the parts which were applicable to our project. We initially conducted a series of interviews with some prospective customers to gather initial information. This information was analyzed by a requirement engineering team (consisting of one business expert and team leaders of various software development teams) in a brainstorming meeting. Then this team conducted several workshops to refine, prioritize old requirements and also gather new ones. All the information collected here was also analyzed by the requirement engineering team in another meeting which helped in grouping the requirements and also developing a high level architectural design of the proposed software system. Workshops and later brainstorming meetings were also part of Structured Brainstorming and Evaluative Survey Technique (SBEST) proposed by Lueke [22]. Also, the requirement engineering process for this market-driven software development was dynamic in that it worked closely with an iterative release cycle as recommended by Sawyer et al. [17]. We did not stop gathering requirements when construction began. It continued even when construction was going on. We used simple natural language to store requirements in a repository. As the software was developed in several releases, we got the feedback from customers after every release. This feedback helped in adding new requirements, refining old ones and also gave an idea about the usability which is a very important attribute of any market driven software to have a competitive advantage in the market as supported by Natt och Dag et. al. [28].

Generally, the case study method is a preferred strategy when "how" and "why" questions are being posed, and the researcher has little control over events [29]. The case study method, a qualitative and descriptive research method, looks intensely at an individual or small participants, drawing conclusions only about the participants or group and only in the specific context [29]. The case study method is an ideal methodology when a holistic, in-depth investigation is required [30].

3
Project Background – A Case Study
Pozadina projekta – Analiza slučaja

The organization is a small and medium enterprise (SME) situated in a software technology park and has been developing software for different domains by using ASD methods for the last six years. This company is engaged in the design, development and implementation of software for various applications/information systems in national and international markets.

A Complex software project (Supply Chain Management – SCM) was developed for the market. There were many potential customers. There were two perspectives of this project. One of them was engineering because of the business area and the solutions types. The other one is software that will provide the solutions as a highly qualified, reliable, accurate and efficiently working software product. The engineering part was based on the operational-research area in the industrial engineering domain. All the optimization problems and solutions related to that topic are defined theoretically. Their results have to be verified by using appropriate tools before they can be transferred as software solutions. Some of the characteristics of the project were:

- The scale project was large
- Project complexity was high
- Acquaintance with the domain was slight
- Initially there was insufficient requirement specification
- Quick release was important to have an edge in the market
- No defined set of customers (There were some prospective customers)
- There were multiple development teams and each team size was small. These teams concurrently developed different parts of SCM software.

The requirement engineering phase for supply chain management software was conducted in several iterations as shown in Fig. 1. According to Jiang et. al. [32], if the project size is large and the project complexity is high, it is better to use a systematic technique to elicit, analyze, document, verify and validate requirements. Initially, we conducted interviews to gather requirements from many potential customers. We tried to observe their problems. Customers talked about their expectations from different areas of the domain. This information was analyzed by a requirement engineering team during brainstorming meetings within the company and this analysis helped in filling the right location in the overall system working scenario. A requirement engineering team consisted of team leaders of various small development teams and one business expert of supply chain management domain. One of the development team members, who had knowledge of this particular domain, played the role of business expert. In those meetings, the business expert within the company pointed out the related problems and their solution that is defined in the concept of the working domain. During these early phases, there was an emphasis on defining a vision and scope, and identifying functions and features at a high level (such as just the names of use cases and features).

Later on workshops were conducted to refine, prioritize and resolve conflicts among requirements. These workshops also helped in determining cross-functional implications that are unknown to individual stakeholders

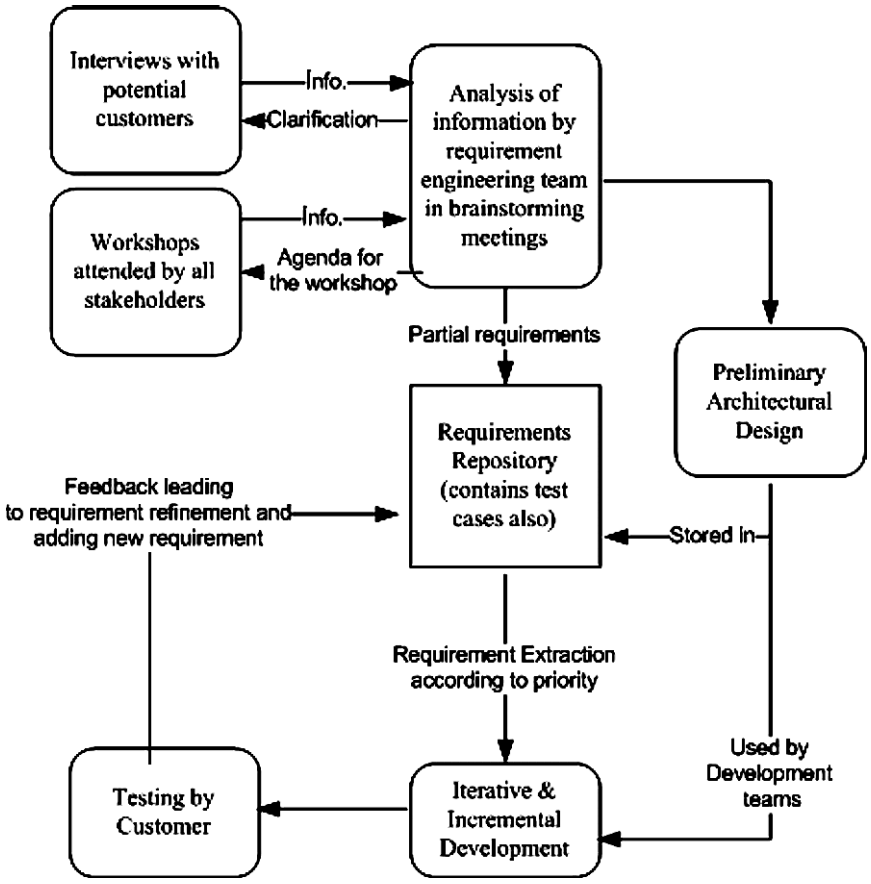

Figure 1 *Requirement Engineering Process for market-driven software development* [31]
Slika 1. *Proces definiranja potreba za tržišno orijentirani softver* [31]

and often missed or defined incompletely during stakeholder interviews. These workshops were held in a series of sessions where each session often only lasted two to four hours, and were often attended by a majority of stakeholders.

These multiple sessions of interviews and workshops helped in obtaining initial requirements (partial requirements) which were used to initiate the development process but the requirement engineering process did not stop here. Interviews and workshops for gathering requirements about parts of the software which were not clear were carried out in parallel with the development of different parts whose requirements were clear. Software requirements were stored in a repository along with their priority. At any point in time we gathered "could do", "should do" and "must do" requirements [33]. These requirements were stored in a centralized repository where they could be viewed, prioritized, and "mined" for iteration features. For each iteration, during the development, according to defined functionality of the iteration, the requirements were selected from that repository according to their priority, their use-cases and working scenarios were prepared by the domain expert and were supplied to the development team.

Requirements were accessible to all team members, available to be enhanced and revised over time, and remained reasonably current to directly drive testing as they were implemented. There were other critical non-functional requirements such as performance, portability, usability, reliability because of the constraints of the business domain. Some of these non-functional requirements were recognized by customers. Others like portability were recognized by the RE team.

Also, the requirement engineering team did the preliminary architectural design during the requirement engineering phase using the initial requirements. This is supported by Mead [34] that architecture modelling and trade studies are important activities that should take place during requirements engineering activities, not after the requirements have been defined. Software architecture must be considered during requirements engineering to ensure that the requirements are valid, consistent, complete, feasible etc.[34]. There were many development teams working concurrently on different parts of the software. To avoid any confusion between these teams and also to have a common picture of what they were developing, RE team designed the core architecture of the system. This was the structure that specified the whole system as major components, modules; collaborations, interactions and interfaces between them and the responsibility of each module. All the defined modules and components were drawn as black boxes and the interactions between them were represented by arrows. Development of each module was assigned to different teams as parallel tasks. The responsibilities and collaborations were defined clearly for each module (i.e. Controller, IO manager) and sites (DBMS, GIS, Application Server). This structure was open to change as a result of customer's feedback from future iterations. Since it was a basis (core) structure and there was collective ownership on that part by the team members, it was important to document the structure in order for it to be accessed easily. The diagrams, responsibilities, functionalities and scenarios were documented by the requirement engineering team. Object-Oriented design techniques were used for architectural design so as to increase applicability of the iterative and incremental

development process because Object-oriented design provides modularity, minimum coupling and maximum cohesion, thus a flexible structure. Ferrett and Offutt [35] also found in their study that object-oriented programs are more modular than procedural programs. Another benefit of using Object-oriented techniques was that it enabled us to define the tasks in parallel, since all modules provide encapsulation and a loosely coupled structure, each could be developed independently as a sub-product and then could be integrated easily because of well-defined interfaces. Once the core was built, team leaders who were a part of the requirement engineering team along with the business expert took active part in the requirement engineering process, returned to their respective teams and the development was done in parallel with multiple teams by using short iterations. Each team leader had a clearer picture and common vision due to the architectural design and could better convey and maintain it for the rest of the project. Moreover, each team leader acted as a liaison to the other teams. Also, after spending some close time with the other team leaders, there was improved communication between these teams. These team leaders played a dual role during the whole project. They took an active part during the requirement engineering process and also they were leading different development teams working in parallel on different parts of the software.

As this project was market-driven, it was not possible to get all the requirements by only conducting interviews and workshops with some potential customers. Also, requirements were not stable. The rate of change in requirements was high so a more flexible approach, like prototyping, needed to be used to gather additional requirements and refine the old ones. Projects with higher requirements volatility require a more flexible approach [32]. Also, the quick release of software was important to have an edge in the highly competitive market so we started developing software with the initial set of requirements by using an iterative and evolutionary approach. These iterations had short timeframes. These evolutionary prototypes of software were used to get feedback from customers. This feedback helped in extracting new requirements and further refinement of previous requirements. Prototypes were suggested as a way to improve the process of defining requirements in market-driven agile-oriented software projects [36].

4
Discussion and lessons learned
Diskusija i stečene spoznaje

As it is accepted nowadays, the product quality is particularly dependent on how requirements engineering practices have been performed [5, 6]. Conventional methodologies are focused on anticipation abilities and can be termed as plan based [37, 38] because these process models are defined in such a way that the later an error is discovered, the more expensive will it be to correct it. As opposed to this, agile methods perceive each change like a chance to improve the system and increase customer satisfaction. One of the principles of ASD is giving the highest priority to achieving customer satisfaction through early and continuous delivery of valuable software [39]. So responding to change over following a plan is one of the agile values [2]. Agile teams do not try to avoid changes but try to understand what is behind them and seek to embrace them [7]. The resulting set of requirements, after a change is introduced, will be evaluated and rated and those requirements that will deliver the highest value to the customer are sought [7]. For instance here in this project at any point in time we have collected a large number of "could do", "should do" and "must do" requirements. Furthermore, these were aggregated in a centralized repository where they could be viewed, prioritized, and "mined" for future iterations. One of the main aims of agile methods is to reduce the cost caused by these changes in requirements simplifying the requirements management and documentation tasks [7]. The definition of tasks related to requirements is very often kept informal in agile approaches. Therefore, although there is evidence of the advantages that agile methodologies provide in small-scale projects, it is still difficult to scale to large projects applying among others the principle responding to change over following a plan [7]. Some lessons learned during this market-driven software project development are:

a) The involvement of some prospective customers is important for the success of the project. Although we usually got the cooperation of customers, sometimes it posed some challenges as well. Sometimes their ideas were entirely different from one another. Furthermore, sometimes at a very crucial moment, they were not able to be present.
b) The presence of a business expert is also significant for the success of the project. In this project, a development team member, who had worked in the past on a similar kind of project, played the role of a business expert. Since this person was not entirely from this particular domain, we could not rely solely on his knowledge and that is why we sought the involvement of some prospective customers. But this business expert played a very crucial role in resolving the conflicts among requirements and also prioritizing these requirements. Also, when the customers could not be present, this member filled that space which we think is quite significant for a market driven project.
c) Preliminary interviews helped in deciding the scope of the problem. Also, they helped in describing the high level description of the requirements.
d) A brainstorming meeting among the requirement engineering team played the role of filter before workshops. Issues which could be resolved without the help of all stakeholders were solved here and therefore saved lots of time. Also, they helped in setting the agenda for the workshops.
e) The role of workshop is very important. They not only helped in refining, prioritizing and resolving conflicts among requirements but also gelled (blend) all stakeholders. This instilled the feeling of a common goal among stakeholders and motivated them to work cohesively towards achieving it.
f) Architectural design, which was done by the requirement engineering team, helped in making a full and clearer picture of the entire system among all different development teams working on different parts of the system.

5 Conclusion
Zaključak

It has been established that Requirement Engineering for market-driven software projects is different than customer-specific software projects. Time-to-market and insufficient initial requirements are two major challenges in market-driven software projects. We handled these challenges by using agile methods for the market-driven software development. Agile methods advocate that software should be released in increments with higher priority requirements implemented in earlier releases and low priority requirements can be excluded to be implemented in a later release. Also, feedback from these releases helps in refining old requirements and adding new ones. Agile methods put emphasis on the dynamic requirement engineering phase which works closely with an iterative release cycle. This process works well for market-driven software projects because it solves two major challenges mentioned above.

Agile methods support gathering requirements in an iterative way as it is impossible to know all the requirements before the development begins. Having a complete set of requirements before construction begins is not a necessity if we use agile methods. In fact, with agility, the requirement engineering phase for market-driven software development can be made dynamic enough to gather and manage requirements from different sources during different timelines of the project.

The product can be released on time (as early as possible) using agility so as to have an edge in the market as time-to-market is another important challenge. This can be achieved by developing software in different releases with high priority requirements implemented in the first release. In the future, this requirement engineering process can be applied to similar kind of projects and thus can be validated after comparing the results from those projects.

Acknowledgement
Zahvala

The authors would like to thank technical editor Prof. Dr. Milan Kljajin and reviewers for helping to improve the paper. We would also like to thank Dr. Ceylan Ertung – Academic Writing & Advisory Center (AWAC) of Atilim University for nicely editing the manuscript.

6 References
Literatura

[1] Srinivasan, J.; Dobrin, R.; Lundqvist, K. 'State of the Art' in Using Agile Methods for Embedded Systems Development. // COMPSAC (2) 2009: 522-527.
[2] Cockburn, A. The End of Software Engineering and the Start of Economic – Cooperative Gaming. // Computer Science and Information Systems, 1, 1 (2004), February, 1-32.
[3] Vijayasarathy, L. R.; Turk, D. Agile software development: A survey of early adopters. // Journal of Information Technology Management 19, 2(2008), 1-8.
[4] Zowghi, D.; Paryani, S. Teaching Requirements Engineering through Role Playing: Lessons Learnt. In Proceedings of the 11th IEEE International Conference on Requirements Engineering, RE Conference. // IEEE Computer Society, 2003, 233-241.
[5] Nuseibeh, B. and Easterbrook, S. Requirements engineering: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM, New York, NY, 35-46.
[6] Cheng, B. H.; Atlee, J. M. Research Directions in Requirements Engineering. In 2007 Future of Software Engineering (May 23-25, 2007). International Conference on Software Engineering. // IEEE Computer Society, 2007, 285-303.
[7] Rodriguez, P.; Yague, A.; Alarcon, P. P.; Garbajosa, J. Some Findings Concerning Requirements in Agile methodologies, F. Bomarius et al. (Eds.) : PROFES 2009, LNBIP 32, pp. 171-184.
[8] Sawyer, P. Packaged Software: Challenges for RE. // Proceedings of the Sixth Int. Workshop on Requirements Engineering: Foundations of Software Quality, Stockholm, Sweden, 2000, pp 137-142.
[9] Karlsson, L.; Dahlstedt, A. G.; Regnell, B.; Natt och Dag, J.; Persson, A. Requirements engineering challenges in market-driven software development - An interview study with practitioners. // Information and Software Technology, Volume 49, Issue 6, Qualitative Software Engineering Research, June 2007, pp 588-604.
[10] Potts, C. Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. // Proceedings of the second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, York, UK, 1995, pp. 128-130.
[11] Cockburn, A.; Highsmith, J. Agile Software Development, the People Factor, Computer, 4, 11(2001), pp 131-133.
[12] Lopez-Nores, M.; Pazos-Arias, J.; Garcia-Duque, J.; Barragans-Martinez, B. An agile approach to support Incremental Development of Requirements Specifications. // Proceeding of the 2006 Australian Software Engineering Conference (ASWEC'06).
[13] Orr, K. Agile Requiremets: Opportunity or Oxymoron? // IEEE Software, 21, 3(2004), 71-73.
[14] Ambler, S. W. Agile Modelling: Effective Practices for Extreme Programming and Unified Process. // John Wiley & Sons, 2002.
[15] Leffingwell, D.; Widrig, D. Managing Software Requirements: a Use Case Approach, John Wiley and Sons, 2003.
[16] Nawrocki, J. R.; Jasinski, M.; Walter, B.; Wojciechowski, A. Extreme Programming Modified: Embrace Requirements Engineering Practices, In 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02), 2002, 303-310.
[17] Sawyer, P.; Sommerville, I.; Kotonya, G. Improving Market-Driven RE Processes. // Proceedings of the International Conference on Product-Focused Software Process Improvement (Profes '99), Oulu, Finland, June 1999, 222-236,
[18] Novorita, R.; Grube, G. Benefits of Structured Requirements Methods for Market-Based Enterprises. // Proceedings of International Council of Systems Engineering, sixth Annual International Symposium on systems Engineering: Practice and Tools (INCOSE 96), Boston, USA, July 1996.
[19] Agerfalk, P.; Fitzgerald, B. Flexible and Distributed Software Processes: Old Petunias in New Bowls? // Communications of the ACM, 10, 49(2006), 27-34.
[20] Rajlich, V. Changing the Paradigm of Software Engineering, Communications of the ACM, 8, 49(2006), 67-70.
[21] Dyba, T.; Dingsoyr, T. Empirical studies of agile software development: A systematic review. Information and Software Technology, 50, 9-10 (Aug. 2008), 833-859.
[22] Lueke, E. Gathering and implementing market-driven requirements. // Professional Communication Conference, 1995. IPCC '95 Proceedings. 'Smooth sailing to the Future'. IEEE International, 27-29 Sept. 1995, 122-126.
[23] Yeh, A. C. Requirements engineering support technique (REQUEST): a market driven requirements management process. // Proceedings of the Second Symposium on assessment of quality software Development Tools. // Los Alamitos, CA: IEEE Computer Society Press., 27-29 May 1992, 211-223.

[24] Tuunanen, T.; Rossi, M., Market driven requirements elicitation via critical success chains. Requirements Engineering Conference. // Proceedings. 11th IEEE International, 8-12 Sept. 2003, 367 – 368.
[25] Regnell, B.; Höst, M.; Natt och Dag, J.; Beremark, P.; Hjelm, T. An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. // Requirements Engineering, 6(2001), 51–62.
[26] Natt och Dag, J.; Regnell, B.; Carlshamre, P.; Andersson, M.; Karlsson, J. A feasibility study of automated natural language requirements analysis in market-driven development. Requirements Engineering, 7(2002), 20–33.
[27] Höst, M.; Regnell, B.; Natt och Dag, J.; Nedstam, J.; Nyberg, C. Exploring bottlenecks in market-driven requirements management processes with discrete event simulations, Journal of Systems and Software, 59(2001), 323-332.
[28] Nattoch, Dag, J.; Regnell, B.; Madsen, O. S.; Aurum, A. An industrial case study of usability engineering in market-driven packaged software development. // Smith, M. J.; Salvendy, G.; Harris, D.; Koubek, R. J. (eds.) Proceedings of HCI International. Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality, Erlbaum, Mahwah,1(2001), 425–429.
[29] Yin, R. K. Case Study Research: Design and Methods, 3rd Edition, Sage Publications, Thousands Oaks, CA, 2003.
[30] Feagin, J.; Orum, A.; Sjoberg, G. (Eds.) A Case for Case Study, University of North Carolina Press, Chapel Hill, NC, 1991.
[31] Mishra, D.; Mishra, A.; Yazici, A. Successful Requirement Elicitation by Combining Requirement Engineering Techniques. IEEE ICADIWT 2008 conference held in VSB-Technical University of Ostrava Czech Republic during Aug. 4-6, 2008, 258-263.
[32] Jiang, L.; Eberlein, A.; Far, B. F. Combining Requirements Engineering Techniques – Theory and Case study. // Proceeding of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05).
[33] Mishra, D.; Mishra, A. Achieving Success in Supply Chain Management Software by Agility. // J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, 237-246.
[34] Mead, N. R.; Shekaran, C.; Garlan, D.; Jackson, M.; Potts, C.; Reubenstein, H. B. The role of software architecture in requirements engineering. // Proceeding of the First International Conference on Requirements Engineering, 18-22 April 1994, 239-245.
[35] Ferrett, L. K.; Offutt, J. An Empirical Comparison of Modularity of Procedural and Object-oriented Software. // Proceedings of the Eighth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'02), 2002, 173-182.
[36] Boness, K.; Harrison, R. Goal Sketching: Towards Agile Requirements Engineering. // Proceedings of the international Conference on Software Engineering Advances (August 25-31, 2007). ICSEA. IEEE Computer Society.
[37] Miller, R. Managing Software or Growth without fear, control and the manufacturing mindset: Addison-Wesley Professional Reading, 2003.
[38] Boehm, B. W. Software Engineering Economics, Prentice-Hall Advances in Computing Science & Technology Series, Prentice Hall PTR, Englewood Cliffs, 1981.
[39] Fowler, M.; Highsmith, J. The Agile Manifesto, Software Development Magazine, 2001. http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm
[40] Mishra, D.; Mishra, A. Market-Driven Software Project through Agility: Requirements Engineering Perspective, W. Abramowicz and D. Flejter (Eds.) : BIS 2009 workshops, LNBIP, 37(2009), 103-112.

Autors' addresses
Adrese autora

Deepti Mishra
Department of Computer Engineering
Atilim University,
Incek Gölbasi, 06836, Ankara, Turkey
deepti@atilim.edu.tr

Alok Mishra
Department of Computer Engineering
Atilim University,
Incek Gölbasi, 06836, Ankara, Turkey
alok@atilim.edu.tr